An experiment about the thread response time - Interim Report -

Katsuya Matsubara

Igel Co., Ltd

Renesas Solutions

2005/9/30

Background

- We want to develop device drivers in the User level
  - Easy development
  - Decrease the risk of system down coursed by driver bug.
  - （GPL**の回避**） …may be it will be better to remain it in Japanese ☺
- Problems
  - Access to I/O Memory, physical devices.
  - Transfer the interrupt request (IRQ)
  - **Reaction speed to the Interrupt request**
  - ・・・・
- **New features in Kernel 2.6**
  - **NPTL(Native POSIX Thread Library)**
  - **Improvement of the scheduler（like O(1) scheduler）**
  - **Kernel Pre-emption**
  - ・・・・

Objectives of the Experiment

- Measure the thread response speed
  - Especially for the waking up time
- Check whether we can utilize the RT thread

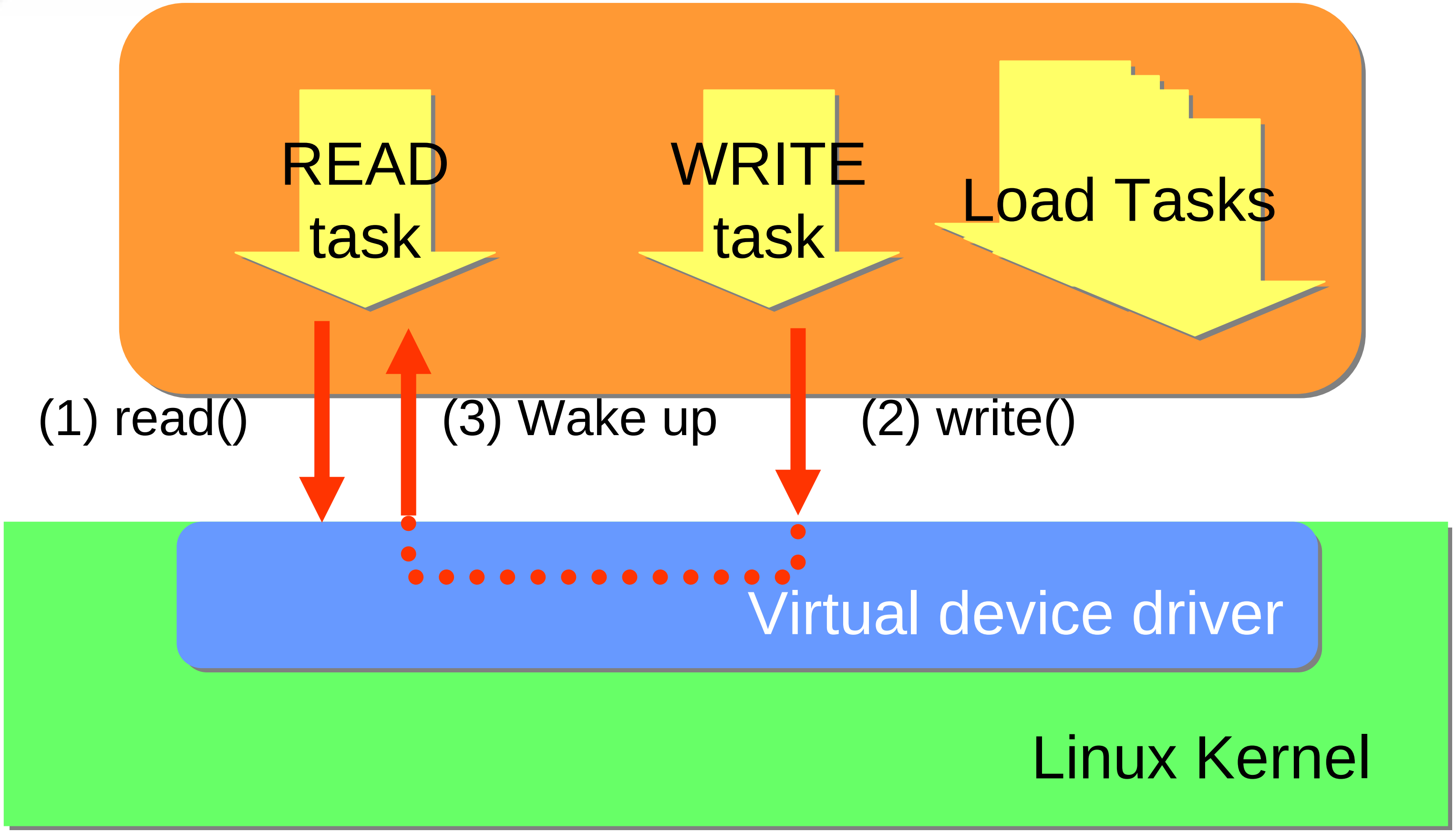
Environment

- Renesas RTS7751R2D (big-endian) solution engine
  - SH4(SH7751) 240MHz
  - 64MB SDRAM
  - 10base-T Ethernet
- Linux 2.6.8.1
- Glibc 2.3.4
- Rootfs is on the NFS Server

Programs for the Experiment

- Virtual device driver
  - Make virtual device file (/dev/irqhook)
  - When any process "write" to the device file, wake up a thread which is blocking as "read" to the device file
- WRITE task
  - Issue a trigger to wake up "read" thread by issuing "write" to the virtual device file
- READ task
  - Issue "read()" to the virtual device file and block it.
- Load task
  - The "rival task" which is also to be the target of the scheduling as same as above tasks.

Programs for the Experiment (contd.)



READ task

```
void read_task () {
    ….

    /* Open virtual device */
    fd = open(“/dev/irqhook”, O_RDONLY);

    for(i=0; i<10000; i++) {
        /* issue read() to block */
        read(fd, &tv1[i], sizeof(struct timeval));
        /* record time */
        gettimeofday(&tv_e[i], NULL);
    }
}
```

WRITE task

```
void write_task () {
    ….

    /* Open virtual device file */
    fd = open(“/dev/irqhook”, O_RDWR);

    ts.tv_sec = 0;
    ts.tv_nsec = 16 * 1000 * 1000; /* 16ms */

    for(i=0; i<10000; i++) {
        /* record time */
        gettimeofday(&tv_s[i], NULL);
        /* issue write() */
        write(fd, &ch, sizeof(char));
        /* Sleep 16ms */
        nanosleep(&ts, NULL);
    }
}
```

Virtual Device Driver

```
ssize_t irqhook_write( …. ) {
    ….
    do_gettimeofday(&tm);

    atomic_inc(&count);
    /* Wake up the task in the
   waiting queue */
    wake_up(&q);
    ….
}
```

```
ssize_t irqhook_read( …. ) {
    ….
    pending = atomic_read(&count);
    while (pending == 0) {
        prepare_to_wait(&q, &wait,
                TASK_INTERRUPTIBLE);
        pending =
              atomic_read(&count);
        /* Sleep until write() */
        if (pending == 0)
                    schedule();
        finish_wait(&q, &wait);
        ….
    }
    copy_to_user(bufp, &tm,
            sizeof(struct timeval));
    ….
}
```

Load task

```
void busyloop() {
    ….
    ts.tv_sec = 0;
    ts.tv_nsec = 1 * 1000; /* 1us */

    while (1) {
        nanosleep(&ts, NULL); /* Wake up in each 1 micro sec
   */
    }
}
```

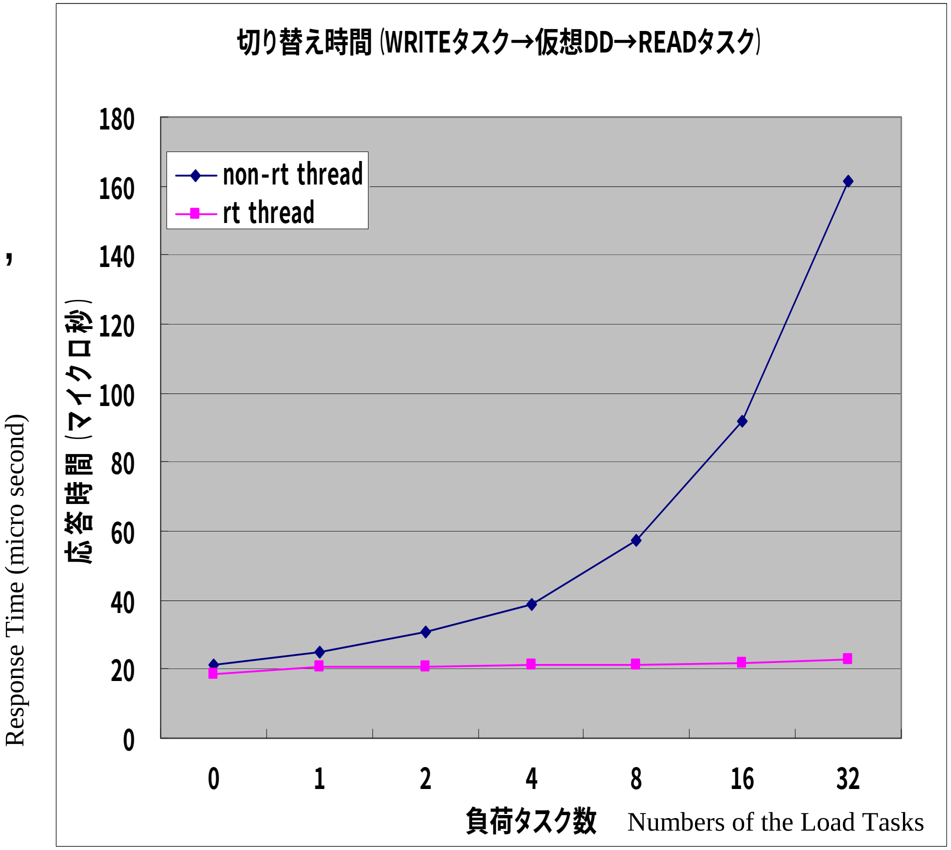
Conditions of the Experiment

- Repeat 10,000 times the READ task and WRITE task, to the virtual device file (issue read() and write()).
- Write Task: issue “write()” in each 16 ms.
- Read Task: issue “read()” just after previous “read()” completed
- Measure the time, from issuing the write() in the Write task, to the Read task returns from read(). At the same time measure the time, from “write()” is accepted by the virtual device driver, to the READ task starts.
- During the measurement, the Load tasks which repeats “nanosleep()” of 1 micro second will be evoked. Measure the case of the number of the Load task, 0,1,2,4,8,16,32.
- Measure two cases on the Read task and the Write task:
  - Execution upon the ordinary (normal?) thread (Non-RT thread).
  - Execution upon the RT thread (Priority 1, Round Robin Scheduling (SCHED_RR))
- Load task(s) is (are) executed on the Non-RT thread

Result
Thread switch over time

- Task execution time is constant even if the number of the load tasks increased
- In case of the non-RT thread, it increases according to the number of the load tasks

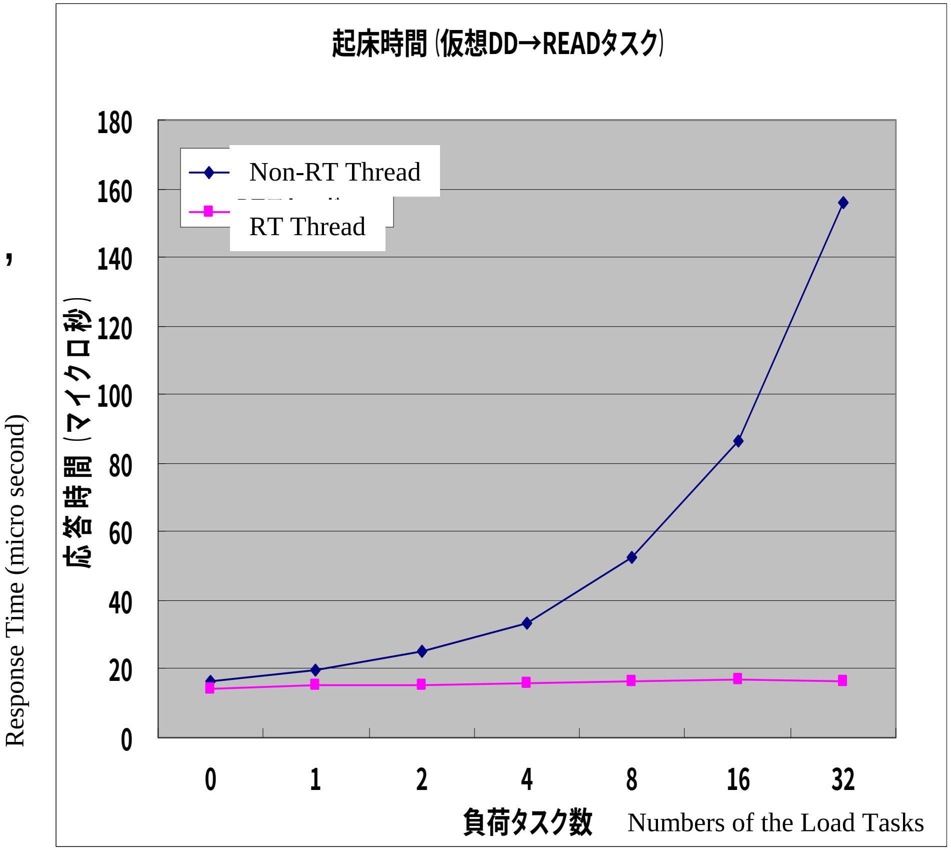
Switch over time (Write Task -> Virtual Device Driver -> Read Task)



Result
Thread waking up time

- RT Thread execution time is constant even if the number of the load tasks increased
- In case of the non-RT thread, thread switching overhead increases according to the wake up time

Wake up time (Virtual D.D. -> Read Task)



Considerations

- RT thread provided constant reaction time. It is because RT thread becomes the target of dispatching immediately whenever it is requested nevertheless the condition of the Non-RT threads such as the Load Task.
- The behavior of Non-RT thread is affected by the other tasks.

RT thread is ideal for developing the user level device drivers,
However how about Non-RT thread???

Behavior of Non-RT thread

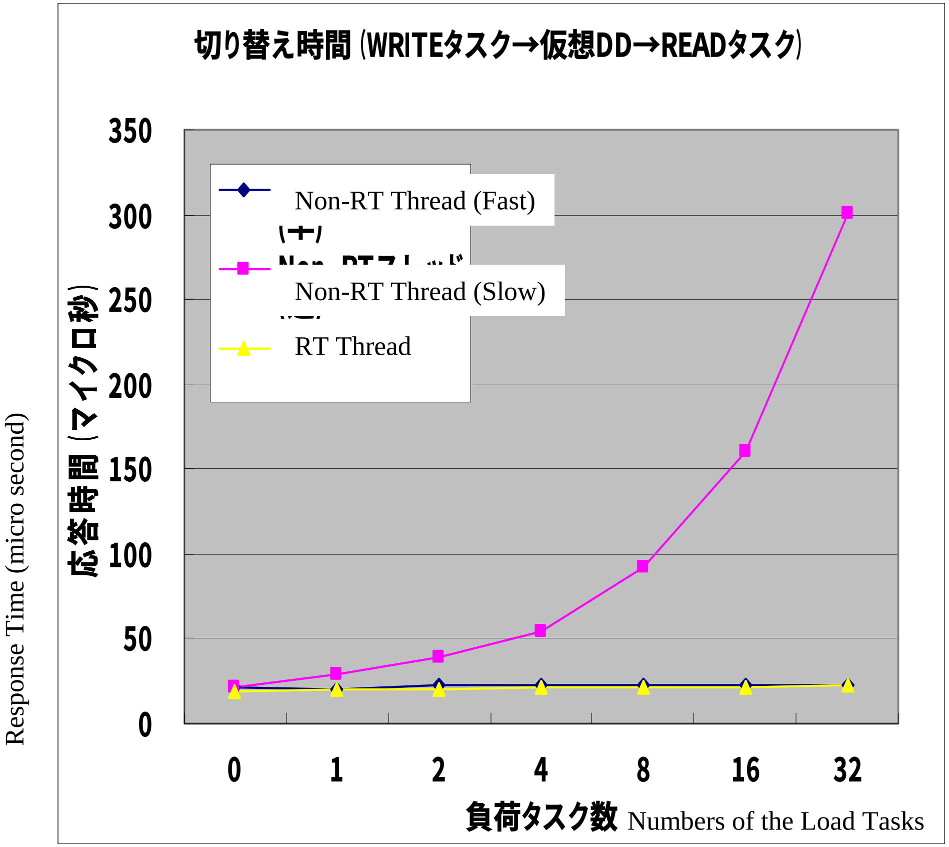
- An interesting (strange) behavior observed in the result of the measurement of Non-RT thread.

Result

Separated Non-RT thread switching time into Fast and Slow

- Non-RT thread response time (fast case) and that of RT thread is almost same.
- Non-RT thread response time (slow case) increase in accordance with the number of the Load Tasks.

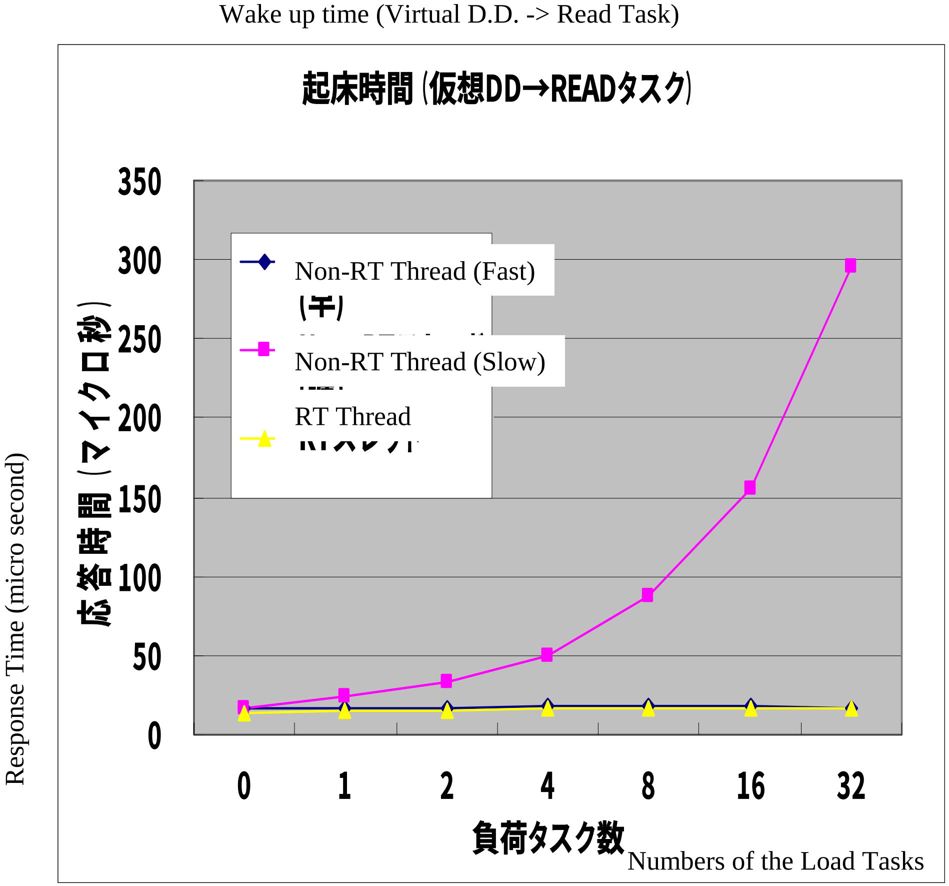
Switch over time (Write Task -> Virtual Device Driver -> Read Task)



Result

Separated Non-RT thread wake up time into Fast and Slow

- Same result as that of switching time



Measurement
Case: Non-RT Thread (Load tasks 16)

Number of Loops	Switching Time	Response time
ループ回数	切り替え時間	応答時間
0	**94**	**185**
1	**17**	**23**
2	**18**	**23**
3	**17**	**21**
4	**17**	**22**
5	**16**	**22**
6	**17**	**22**
7	155	160
8	17	22
9	154	159
10	17	22
11	154	160
‥	‥	‥

Measurement

Case: RT Thread (Load tasks 16)

Number of Loops	Switching Time	Response time
ループ回数	切り替え時間	応答時間
0	**91**	**163**
1	**17**	**23**
2	**16**	**21**
3	**16**	**21**
4	**17**	**21**
5	**16**	**21**
6	**16**	**20**
7	**16**	**21**
8	**16**	**20**
9	**16**	**21**
10	**16**	**20**
11	**16**	**21**
‥	‥	‥

QUESTION???

- The response time of the Non-RT thread is stable in the initial few loops.
- However, after then, why the response time repeat “Fast” and “Slow” alternately?
- Why the first response time is so slow in both case that RT exists and not exists.

Discussion

- How do you think about it???
  - “It must be caused by….!”
  - “How about to examine the other way such as…!”
  - “This tool will be effective for this experiment!”
  - “Your experiment may have problem on…!”
  - Etc.etc.etc……

I think…

- Why the wake up time becomes slow and fast alternately?
  - Something courses to make both priority on Load task and Read task to be same. (?)
  - Once it wake up, then, when it listed into the run queue, it added after the Load task. (?)
  - Once it delayed, because of the longer sleep time, it will be dispatched upon the condition of the higher priority. (?)
- Why it is stable in the first few times?
  - ?
- Why it is extremely slow at the first time?
  - ??